VEBAV - A Simple, Scalable and Fast Authorship Verification Scheme
Notebook for PAN at CLEF 2014

Oren Halvani[⋆] and Martin Steinebach

Fraunhofer Institute for Secure Information Technology SIT
Rheinstrasse 75, 64295 Darmstadt, Germany
{FirstName.LastName}@SIT.Fraunhofer.de

Abstract We present VEBAV - a simple, scalable and fast authorship verification scheme for the Author Identification (AI) task within the PAN-2014 competition. VEBAV (VEctor-Based Authorship Verifier), which is a modification of our existing PAN-2013 approach, is an intrinsic one-class-verification method, based on a simple distance function. VEBAV provides a number of benefits as for instance the independence of linguistic resources and tools like ontologies, thesauruses, language models, dictionaries, spellcheckers, etc. Another benefit is the low runtime of the method, due to the fact that deep linguistic processing techniques like POS-tagging, chunking or parsing are not taken into account. A further benefit of VEBAV is the ability to handle more as only one language. More concretely, it can be applied on documents written in Indo-European languages such as Dutch, English, Greek or Spanish. Regarding its configuration VEBAV can be extended or modified easily by replacing its underlying components. These include, for instance the distance function (required for classification), the acceptance criterion, the underlying features including their parameters and many more. In our experiments we achieved regarding a 20%-split of the PAN 2014 AI-training-corpus an overall accuracy score of 65,83% (in detail: 80% for Dutch-Essays, 55% for Dutch-Reviews, 55% for English-Essays, 80% English-Novels, 70% for Greek-Articles and 55% for Spanish-Articles).

Keywords: Authorship verification, one-class-classification

1 Introduction

Authorship Verification (AV) is a sub-discipline of Authorship Analysis [4, Page: 3], which itself is an integral part of digital text forensics. It can be applied in many forensic scenarios as for instance checking the authenticity of written contracts, threats, insults, testaments, etc. where the goal of AV remains always the same: Verify if two documents $\mathcal{D}_{\mathcal{A}}$ and $\mathcal{D}_{\mathcal{A}_?}$ are written by the same author $\mathcal{A}$, or not. An alternative reformulation of the goal is to verify the authorship of $\mathcal{D}_{\mathcal{A}_?}$, given a set of sample documents of $\mathcal{A}$. From a Machine Learning perspective AV clearly forms an one-class-classification problem [3], due to the fact that $\mathcal{A}$ is the only target class to be distinguished among all other possible classes (authors), where their number can be theoretically infinite.

[⋆] Corresponding author.

In order to perform AV at least four components are mandatorily required:

- The document $\mathcal{D}_{\mathcal{A}_?}$, which should be verified regarding its alleged authorship.
- A training set $\mathbb{D}_{\mathcal{A}} = \{\mathcal{D}_{1_{\mathcal{A}}}, \mathcal{D}_{2_{\mathcal{A}}}, ...\}$, where each $\mathcal{D}_{i_{\mathcal{A}}}$ represents a sample document of $\mathcal{A}$.
- A set of features $\mathbb{F} = \{f_1, f_2, ...\}$, where each f_j (*style marker*) should help to model the writing style of $\mathcal{D}_{\mathcal{A}_?}$ and each $\mathcal{D}_{i_{\mathcal{A}}} \in \mathbb{D}_{\mathcal{A}}$.
- At least one classification method, which accepts or rejects the given authorship based on $\mathbb{F}$ and a predefined or dynamically determined threshold θ.

The aim of this paper is to provide a simple, scalable and fast AV scheme, which offers a number of benefits as for instance promising detection rates, easy implementation, low runtime, independence of language or linguistic resources as well as easy modifiability and expandability. Our proposed AV scheme, denoted by `VEBAV`, is based on our earlier approach regarding the PAN 2013 AI-task, which itself formed a modification of the Nearest Neighbor (NN) one-class classification technique, described by Tax in [5, Page: 69]. In a nutshell, `VEBAV` takes as an input a set of sample documents of a known author ($\mathbb{D}_{\mathcal{A}}$) and one document of an unknown author ($\mathcal{D}_{\mathcal{A}_?}$). All documents in $\mathbb{D}_{\mathcal{A}}$ are first concatenated to a big document which is then splitted again into (near) eqal-sized chunks (such that $\mathbb{D}_{\mathcal{A}}$ includes now only these chunks). Afterwards, feature vectors are constructed from each $\mathcal{D}_{i_{\mathcal{A}}} \in \mathbb{D}_{\mathcal{A}}$ and from $\mathcal{D}_{\mathcal{A}_?}$ according to preselected feature sets. Next, a representative is selected among the training feature vectors (based on two options), which is important to determine the decision regarding the alleged authorship. In the next step distances are calculated between the representative and the remaining training feature vectors, as well as between the representative and the test feature vector. Depending on all calculated distances a twofold decision regarding the alleged authorship is generated, which includes a ternary decision $\delta \in \{\textit{Yes}, \textit{No}, \textit{Unanswered}\}$ and a probability score ρ that describes the soundness of δ.

The rest of this paper is structured as follows. In the following section we explain how we extract features from the corresponding linguistic layers and which features have been used in our approach. Afterwards, we describe in section 3 our verification scheme. In section 4 we present the corpus that was used to evaluate our scheme, while in section 5 we show our evaluation results. Finally, we draw our conclusions and the challenges we were faced in section 6 and provide some ideas for future work.

2 Features

Features (style markers) are the core of AV, since they are able to approximate writing styles and thus, can help to judge automatically if two texts have a very similar style. If this holds, it is an indicator that both texts could be written by the same author. In the next subsections we explain which *sources* exactly quantifiable features can be retrieved from, which tools are required for this extraction process and also what kind of feature sets we used in our approach.

2.1 Linguistic layers

From a linguistic point of view, features are extracted from so-called linguistic layers that can be understood as abstract units within a text. In summary, the most important linguistic layers are the following:

- **Phoneme layer:** This layer includes phoneme-based features as for example vowels, consonants or also the more sophisticated supra-segmental or prosodic features. Such features can typically be won out of texts by using (pronouncing) dictionaries (e.g. IPA).

- **Character layer:** This layer includes character-based features as for instance prefixes, suffixes or letter n-grams, which typically are extracted from texts via regular expressions.

- **Lexical layer:** This layer includes token-based features as for instance function words or POS-Tags (Part-Of-Speech Tags). These features can be extracted from texts via tokenizers (which often are based on simple regular expressions).

- **Syntactic layer:** This layer includes syntax-based features as for instance constituents (e.g. nominal phrases) or collocations. Such features can be extracted by sophisticated regular expressions or by natural language processing tools (e.g. POS-Tagger). However, the latter one is normally bounded to a specific language model and thus, cannot scale to multiple languages. Besides this the runtime of natural language processing tools is much more higher as the runtime caused by pattern matching via regular expressions.

- **Semantic Layer:** This layer includes semantic-based features, e.g. semantic relations (hyponymous, synonymys, meronyms, etc.). Such features require deep linguistic processing, which often rely on external knowledge resources (e.g. WordNet) or complex tools (e.g. parsers, named entity eecognizers, etc.).

In VEBAV we only make use of the *Character*, *Lexical* and *Syntactic* linguistic layers, due to the effectiveness of their underlying features and the low runtime, caused by the feature extraction process via regular expressions.

2.2 Feature sets

In this paper we use the term *feature set*, denoted by F, which represents features belonging to one or more linguistic layers. In Table 1 we list 14 feature sets that have been used in our experiments. Here, one should pay attention that F_{12}, F_{13} and F_{14} form mixtures of existing feature sets. The idea behind it was to see if such mixtures can outperform[1] single feature sets when it comes to classification.

[1] This was in fact the case, as can be observed in the later experiments (section 5.3).

Table 1. All feature sets used in our approach.

F_i	**Feature set**	**Description**	**Examples**
F_1	*Characters*	All kind of characters	{a,b,1,8,#,<,%,!,...}
F_2	*Letters*	All kind of letters	{a,b,α,β,ä,ß,ó,á,ñ,...}
F_3	*Punctuation marks*	Symbols that structure sentences	{.,:,;,-,",(,),...}
F_4	*Word k Prefixes*	The k starting letters of words	example $\rightsquigarrow$ {e,ex,exa,exam,...}
F_5	*Word k Suffixes*	The k ending letters of words	example $\rightsquigarrow$ {e,le,ple,mple,...}
F_6	*Character n-grams*	Overlapping character-fragments	ex-ample $\rightsquigarrow$ {ex-,x-a,-am,...}
F_7	*Letter n-grams*	Overlapping letter-fragments	ex-ample $\rightsquigarrow$ {exa,xam,amp,...}
F_8	*Tokens*	Segmented character-based units	A [sample] text! $\rightsquigarrow$ {A,[sample],text!}
F_9	*Words*	Segmented letter-based units	A [sample] text! $\rightsquigarrow$ {A,sample,text}
F_{10}	*Token n-grams*	Overlapping token-fragments	Wind and rain! $\rightsquigarrow$ {Wind and, and rain!}
F_{11}	*Word n-grams*	Overlapping word-fragments	Wind and rain! $\rightsquigarrow$ {Wind and, and rain}
F_{12}	*Mix*$_1$	A mix of three feature sets	$F_1 \cup F_3 \cup F_6$
F_{13}	*Mix*$_2$	A mix of three feature sets	$F_1 \cup F_2 \cup F_5$
F_{14}	*Mix*$_3$	A mix of four feature sets	$F_3 \cup F_4 \cup F_5 \cup F_8$

2.3 Parameters

As can be seen in Table 1, six feature sets can be parameterized by n (n-gram sizes) and/or k (length of prefixes/suffixes). It should be emphasized that adjusting these two parameters can influence the results massively. Hence, we generalized both settings by using constant values ($n = k = 3$), learned from earlier experiments, which turned out to be reliable also in the current experiments among the diverse PAN subcorpora.

3 Proposed Verification Scheme

In this section we give a detailed description of our AV scheme VEBAV. For overview reasons we divided the entire workflow of the algorithm into six subsections, where we first explain what kind of preprocessing we perform on the data. The other five subsections focus on the algorithm itself.

3.1 Preprocessing

In contrast to our approach in PAN-2013 we decided in our current approach neither to apply normalization nor noise reduction techniques. Instead, we treat each text as it is, which turned out to be not only less burdensome but also promising. Our only preprocessing mechanism is restricted to concatenate all $\mathcal{D}_{i\mathcal{A}} \in \mathbb{D}_{\mathcal{A}}$ to a single document $\mathcal{D}_{\mathcal{A}}$ which, depending on the length, is splitted again into ℓ (near) equal-sized chunks $\mathcal{D}_{1\mathcal{A}}, \mathcal{D}_{2\mathcal{A}}, ..., \mathcal{D}_{\ell\mathcal{A}}$. Note that in our experiments ℓ is statically set to five chunks if, and only if, the length of $\mathcal{D}_{\mathcal{A}}$ is above 15,000 characters, otherwise ℓ is statically set to three chunks. The reason for this decision is that we observed quite better results in contrast by using only one fixed ℓ for both, short or long texts.

3.2 Vocabulary Generation

In order the form a basis for the construction of feature vectors, we need to build a global feature vocabulary, denoted by $\mathcal{V}$. But beforehand, we first need to select at least one (*appropriate*) feature set F, to know which sort of features should be taken into account. Here, *appropriate* refers to features that...

- ... are able to model high similarity between $\mathcal{D}_{\mathcal{A}_?}$ and $\mathcal{D}_{\mathcal{A}}$ (for the case $\mathcal{A}_? = \mathcal{A}$).
- ... are able to discriminate well between the writing style of $\mathcal{A}$ and all other possible authors (for the case $\mathcal{A}_? \neq \mathcal{A}$).
- ... appear in all generated vocabularies.

Afterwards, we construct from $\mathcal{D}_{\mathcal{A}_?}$ and each chunk $\mathcal{D}_{i\mathcal{A}} \in \mathbb{D}_{\mathcal{A}}$ the corresponding feature vocabularies $\mathcal{V}_{\mathcal{D}_{\mathcal{A}_?}}$ and $\mathcal{V}_{\mathcal{D}_{1\mathcal{A}}}, \mathcal{V}_{\mathcal{D}_{2\mathcal{A}}}, ..., \mathcal{V}_{\mathcal{D}_{\ell\mathcal{A}}}$. As a last step we apply an intersection among all constructed vocabularies to build the global vocabulary $\mathcal{V}$:

$$\mathcal{V} = \mathcal{V}_{\mathcal{D}_{\mathcal{A}_?}} \cap \mathcal{V}_{\mathcal{D}_{1\mathcal{A}}} \cap \mathcal{V}_{\mathcal{D}_{2\mathcal{A}}} \cap ... \cap \mathcal{V}_{\mathcal{D}_{\ell\mathcal{A}}} \tag{1}$$

Note that in `VEBAV` we consider case-sensitive features, such that *word* $\in \mathcal{V}$ and *Word* $\in \mathcal{V}$ are both valid.

3.3 Constructing feature vectors

Once $\mathcal{V}$ is generated, the next step is to construct the feature vectors $\mathcal{F}_{1\mathcal{A}}, \mathcal{F}_{2\mathcal{A}}, ..., \mathcal{F}_{\ell\mathcal{A}}$ from each $\mathcal{D}_{i\mathcal{A}} \in \mathbb{D}_{\mathcal{A}}$ and $\mathcal{F}_{\mathcal{A}?}$ from $\mathcal{D}_{\mathcal{A}_?}$. The construction process regarding these vectors is very straightforward. We look up for each $f_i \in \mathcal{V}$ how often it occurs in some document $\mathcal{D}$ and denote its absolute frequency by α_i. Next, we normalize α_i by the length of $\mathcal{D}$, to obtain its relative frequency, which represents a number $x_i \in [0\,;\,1]$. Hence, the feature vector representation regarding $\mathcal{D}$ is formally defined by:

$$\mathcal{F} = (x_1, x_2, ..., x_n)\,, \text{with } n = |\mathcal{V}|\;. \tag{2}$$

3.4 Representative Selection

After constructing all feature vectors, a representative (training-) feature vector $\mathcal{F}_{rep}$ must be selected. This step is essential for the later determination of the decision regarding the alleged authorship. In general, `VEBAV` offers two options to select $\mathcal{F}_{rep}$:

1. Selecting $\mathcal{F}_{rep}$ dynamically by using a similarity function (e.g. cosine similarity) between all training feature vectors. Here, $\mathcal{F}_{rep}$ is selected according to the feature vector, who is mostly dissimilar from the others. In other terms $\mathcal{F}_{rep}$ can be understand as an outlier.

2. Selecting $\mathcal{F}_{rep}$ manually (static).

Note that in our experiments we chose the first option.

3.5 Distances Calculations

In this step we calculate the distances, needed for the decision determination regarding the alleged authorship. Concretely, we calculate the distances $d_1, d_2, ..., d_\ell$ between $\mathcal{F}_{rep}$ and each $\mathcal{F}_{1\mathcal{A}}, \mathcal{F}_{2\mathcal{A}}, ..., \mathcal{F}_{\ell\mathcal{A}}$ as well as the the distance $d_?$ between $\mathcal{F}_{rep}$ and $\mathcal{F}_{\mathcal{A}?}$.

The calculation of these distances requires a predefined distance function dist(X, Y) with $X = \mathcal{F}_{rep}$ and $Y \in \{\mathcal{F}_{1\mathcal{A}}, \mathcal{F}_{2\mathcal{A}}, ..., \mathcal{F}_{\ell\mathcal{A}}\}$. We implemented a broad range of distance functions in `VEBAV`, where the majority have been taken from [1]. However, for complexity reasons we used only the *Minkowski* distance function in our experiments, which is defined formally as:

$$\text{dist}(X, Y) = \Big(\sum_{i=1}^{n} |x_i - y_i|^{\lambda}\Big)^{\frac{1}{\lambda}}, \text{with } \lambda \in \mathbb{R}_{+} \setminus \{0\}\ , \tag{3}$$

As a last step we calculate the average regarding the distances between $\mathcal{F}_{rep}$ and each $\mathcal{F}_{1\mathcal{A}}, \mathcal{F}_{2\mathcal{A}}, ..., \mathcal{F}_{\ell\mathcal{A}}$ as follows:

$$d_{\varnothing} = \frac{1}{\ell}(d_1 + d_2 + \ldots + d_{\ell}) \tag{4}$$

3.6 Decision Determination

The goal of this step is to construct a twofold decision regarding the alleged authorship, which consists of a ternary decision $\delta \in \{\textit{Yes}, \textit{No}, \textit{Unanswered}\}$ and a probability score ρ. In order to calculate these terms both values are required $d_?$ and $d_{\varnothing}$. For the latter one we use the following adjusted form: $d' = d_{\varnothing} + (\omega \cdot \tau)$, where ω denotes a weight and τ a tolerance parameter, calculated from the a standard deviation of $d_1, d_2, ..., d_{\ell}$. The idea behind ω and τ is to cope with the presence of noisy writing styles, that might be the result of mixing different sample documents of $\mathcal{A}$ together in his training set $\mathbb{D}_{\mathcal{A}}$. Note that in our experiments we chose a neutral weight ($\tau = 1$), since we could not investigate an optimal value for it (even adjustig it by 0.1 can cause a massive drop regarding the classification results). With these we first define the probability score as:

$$\rho = \frac{1}{1 + \frac{d_?}{d'}} \tag{5}$$

Next, we define the ternary decision as follows:

$$\delta = \begin{cases} \textit{Yes}, & d_? < d' \\ \textit{No}, & d_? > d' \\ \textit{Unanswered}, & (d_? = d') \vee (|d_? - d'| < \varepsilon) \end{cases} \tag{6}$$

The concrete semantic behind δ is:

- *Yes*: `VEBAV` accepts the alleged author as the true author ($\mathcal{A}_? = \mathcal{A}$).
- *No*: `VEBAV` rejects the alleged author as the true author ($\mathcal{A}_? \neq \mathcal{A}$).
- *Unanswered*: `VEBAV` is unable to generate a prediction because $d_?$ and d' are equal/near-equal or due to another unexpected result. In any case ρ is set to 0.5. Note that depending on how ε was chosen (regarding the case that $d_?$ and d' are near-equal) the number of *Unanswered* decisions, in the context of a corpus evaluation, can vary considerably. In our experiments we chose $\varepsilon = 0,001$ as this small value restricts the number of *Unanswered* decisions, where ρ is near 0.5.

4 Used Corpora

Regarding our experiments we used the official *PAN-2014 Author Identification* training corpus, denoted by $\mathcal{C}_{\textit{PAN-14}}$, which has been released[2] by the PAN organizers on April 22, 2014. $\mathcal{C}_{\textit{PAN-14}}$ consists of 695 problems (in total 2,382 documents), equally distributed regarding true/false authorships. A problem p_i forms a tuple $(\mathbb{D}_{\mathcal{A}_i}, \mathcal{D}_{\mathcal{A}_i?})$, where $\mathbb{D}_{\mathcal{A}_i}$ denotes the training set of $\mathcal{A}_i$ and $\mathcal{D}_{\mathcal{A}_i?}$ the questioned document, which is (or not) written by $\mathcal{A}_i$. Each problem belongs to one of four languages (*Greek*, *Spanish*, *Greek* and *Spanish* and to one of four genres (*Essays*, *Reviews*, *Novels* and *Articles*). For simplification reasons, $\mathcal{C}_{\textit{PAN-14}}$ is divided into six subcorpora and thus, can be formulated as $\mathcal{C}_{\textit{PAN-14}} = \{\mathcal{C}_{DE}, \mathcal{C}_{DR}, \mathcal{C}_{EE}, \mathcal{C}_{EN}, \mathcal{C}_{GR}, \mathcal{C}_{SP}\}$. This makes it easier to treat each subcorpus independently (e.g. in terms of parameterizations). The full name of each $\mathcal{C} \in \mathcal{C}_{\textit{PAN-14}}$ is given in Table 2.

Table 2. Full names of all subcorpora within the $\mathcal{C}_{\textit{PAN-14}}$ corpus.

$\mathcal{C}_{DE}$: *Dutch-Essays*	$\mathcal{C}_{EE}$: *English-Essays*	$\mathcal{C}_{GR}$: *Greek-Articles*
$\mathcal{C}_{DR}$: *Dutch-Reviews*	$\mathcal{C}_{EN}$: *English-Novels*	$\mathcal{C}_{SP}$: *Spanish-Articles*

For our experiments we used a 80% portion of $\mathcal{C}_{\textit{PAN-14}}$ for training/parameter learning (denoted by $\mathcal{C}_{\textit{PAN-Train}}$), while the remaining 20% portion was used for testing (denoted by $\mathcal{C}_{\textit{PAN-Test}}$). Regarding $\mathcal{C}_{\textit{PAN-Test}}$ we used the same structure as $\mathcal{C}_{\textit{PAN-14}}$ such that $\mathcal{C}_{\textit{PAN-Test}} = \{\mathcal{C}_{DE}, \mathcal{C}_{DR}, \mathcal{C}_{EE}, \mathcal{C}_{EN}, \mathcal{C}_{GR}, \mathcal{C}_{SP}\}$ holds, where the number of problems in each $\mathcal{C} \in \mathcal{C}_{\textit{PAN-Test}}$ equals 20% of the problems in each $\mathcal{C} \in \mathcal{C}_{\textit{PAN-Train}}$. The concrete statistic including the distribution of true/false authorships is given in Table 3.

Table 3. Statistics of $\mathcal{C}_{\textit{PAN-Test}}$.

$\mathcal{C}$	$\lvert\mathcal{C}\rvert$	**Distribution of true/false authorships**
$\mathcal{C}_{DE}$	20	9 true cases / 11 false cases
$\mathcal{C}_{DR}$	20	11 true cases / 9 false cases
$\mathcal{C}_{EE}$	40	2 true cases / 38 false cases
$\mathcal{C}_{EN}$	20	10 true cases / 10 false cases
$\mathcal{C}_{GR}$	20	11 true cases / 9 false cases
$\mathcal{C}_{SP}$	20	10 true cases / 10 false cases

5 Evaluation

In this section we carry out our evaluation regarding the $\mathcal{C}_{\textit{PAN-14}} = \mathcal{C}_{\textit{PAN-Train}} \cup \mathcal{C}_{\textit{PAN-Test}}$ corpus. First we explain which performance measures were used and secondly, how

[2] The corpus can be downloaded from: `http://pan.webis.de` (accessed on June 26, 2014).

the most important parameters were learned from $\mathcal{C}_{\textit{PAN-Train}}$. Finally, we evaluate our approach on $\mathcal{C}_{\textit{PAN-Test}}$.

5.1 Performance measures

In order to evaluate our approach, we used several performance measures, sharing the following variables:

$$m = \textit{Number of problems in } \mathcal{C} \tag{7}$$

$$m_c = \textit{Number of correct answers per } \mathcal{C} \tag{8}$$

$$m_u = \textit{Number of unanswered problems answers per } \mathcal{C} \tag{9}$$

The first performance measure is $\textit{Accuracy} = \frac{m_c}{m}$, whereas for the second performance measure we first need to define two terms:

$$\textit{AUC} = \frac{1}{m}\sum_{i=1}^{m} \rho_i \ , \qquad c@1 = \frac{1}{m}\left(m_c + \left(m_u \cdot \left(\frac{m_c}{m}\right)\right)\right) \tag{10}$$

Here, ρ_i denotes the probability score regarding its corresponding problem p_i, which was defined in section 4. With these two measures we define the second performance measure: $\textit{AUC} \cdot c@1$. Note that for parameter learning from the $\mathcal{C}_{\textit{PAN-Train}}$ corpus we only used the *Accuracy* measure, as (in our opinion) this measure is better interpretable.

5.2 Experiment I: Finding an optimal λ for the *Minkowski* distance function

The intention behind this experiment was to find an optimal λ parameter, used by the *Minkowski* distance function. Since λ has a strong influence on the classification result it must be well-chosen, in order to generalize across the range of all involved corpora and all feature sets. To achieve this generalization we merged all training subcorpora, such that $\mathcal{C}_{\textit{PAN-Train}} = \mathcal{C}_{\textit{DE}} \cup \mathcal{C}_{\textit{DR}} \cup \mathcal{C}_{\textit{EE}} \cup \mathcal{C}_{\textit{EN}} \cup \mathcal{C}_{\textit{GR}} \cup \mathcal{C}_{\textit{SP}}$ holds. Afterwards, we applied `VEBAV` on $\mathcal{C}_{\textit{PAN-Train}}$, where as an input we used all mentioned feature sets in Table 1 and the following 14 predefined λ values $\{0.2, 0.4, 0.6, 0.8, 1, 2, ..., 10\}$. We constructed from the results a table, where the rows represent the feature sets $F_1, F_2, \ldots, F_{14}$, while the columns represent the 14 λ values. Next, we derived a row from this table that includes the medians regarding all columns. This row is illustrated in Figure 1. As can be seen in Figure 1, an optimal range for λ is [0.2; 1], where 0.6 seems to be the most promising one in terms of robustness, among all involved feature sets. As a consequence of this analysis, we decided to use $\lambda = 0.6$ for the further experiments.

5.3 Experiment II: Determinig the classification stregth of all feature sets

In this experiment we wanted to compare the classification stregth of all involved feature sets. For this we applied `VEBAV` on $F_1, F_2, \ldots, F_{14}$ regarding the six subcorpora in $\mathcal{C}_{\textit{PAN-Train}}$, where this time we used the fixed setting $\lambda = 0.6$. From the resulting table (rows = feature sets, columns = subcorpora) we calculated for each row (containing

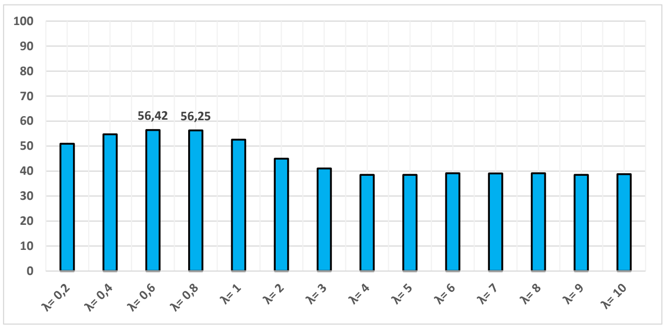
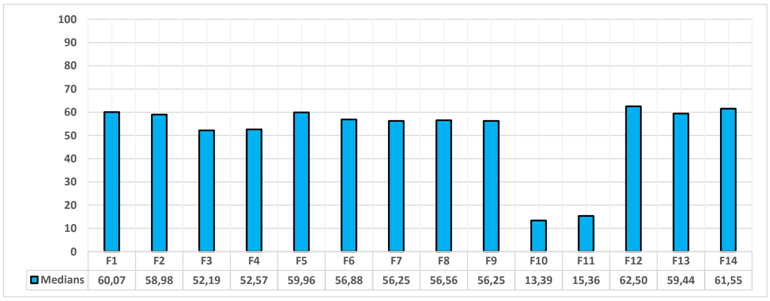


Figure 1. Comparison of different λ values for the *Minkowski* distance function.

the classification results regarding all six subcorpora) the median, which gave us a new column, illustrated in Figure 2. Here, it can be seen that the majority of all feature sets are more or less equally strong, excepting F_{10} and F_{11}, which seem to be useless for VEBAV (at least for $\mathcal{C}_{\textit{PAN-Train}}$). Furthermore, it can be observed that using mixed feature sets lead to slightly better classification results, compared with the majority of the non-mixed feature sets.



	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14
🟦 Medians	60,07	58,98	52,19	52,57	59,96	56,88	56,25	56,56	56,25	13,39	15,36	62,50	59,44	61,55

Figure 2. Comparison of the 14 feature sets across all training corpora (using $\lambda = 0.6$).

In order to get a better picture of how VEBAV performs without considering the medians among the six subcorpora, we show in Figure 3 a comparison of the top three performing feature sets for each individual subcorpus. One interesting observation that can be concluded from Figure 3 is that the feature set F_1 (*characters*) is almost as strong as the mixed feature sets, which involve more sophisticated features (such as *character n-grams* or *tokens*). This shows that by using only characters as features, it is possible to verify authorships with a classification result, which is even better as a random guess (50%). Another observation, which is worth to be mentioned, is the fact that the greek corpus seems to contain problems that are more difficult to judge, compared to the problems in the other subcorpora, where the classification results are obviously better.

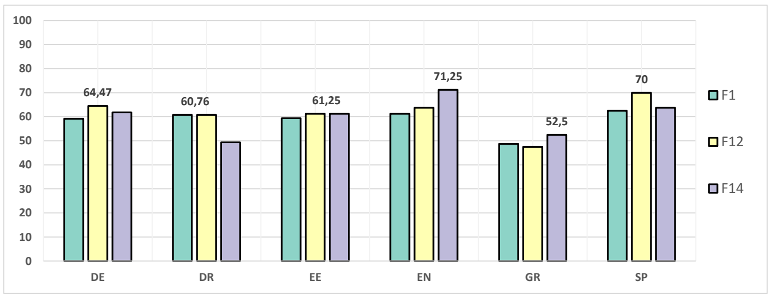


Figure 3. Comparison of the top three performing feature sets (without applying medians among the six subcorpora).

We believe that this is not a language-based issue, but more an effort of the organizers to make the task more challenging, as this was also the case in PAN-2013 [2].

5.4 Experiment III: Single feature sets vs. feature set combinations

In this experiment we were curious to know, if using combinations of feature sets by applying majority-voting can outperform classifications based only on single feature sets. As a setup for this experiment, we picked out the six most promising feature sets $\{F_1, F_2, F_5, F_{12}, F_{13}, F_{14}\}$ and used them to construct a power set $\mathbb{P}$, which includes $2^6 = 64$ feature set combinations. Next, we removed those subsets $F_{comb\,i} \in (\mathbb{P} \setminus \emptyset)$ comprising of an even number of feature set combinations, to enable a fair (non-random based) majority-voting and also to speed the classification process up a little by avoiding unnecessary runs. This led to $2^5 = 32$ suitable combinations $F_{comb\,1}, F_{comb\,2}, ..., F_{comb\,32}$, where we applied each F_{comb} as an input for VEBAV regarding $\mathcal{C}_{\textit{PAN-Train}}$. Next, we stored all combinations and their corresponding classification results in a list (sorted in descending order) and selected the top five combinations, listed in Table 4. Unfortunately, it can be observed from the comparison between Table 4 and Figure 2 that applying majority-voting on feature set combinations gives only negligible improvements ($\approx$ 1-2%) for the most cases. When focussing on F_{12} in Figure 2 it can be even seen that its median accuracy of 62.50% outperfroms any sort of feature set combination. One possible reason for the unsatisfactory results could be the fact that many single feature sets made identical ternary decisions (*Yes*, *No*, *Unanswered*), such that applying majority-voting is only effectively in few cases. We observed this phenomenon several times by stepping through the code, using the debugger. Hence, we do not consider the usage of feature set combinations for further evaluations in this paper.

5.5 Experiment IV: Obtaining corpus dependent parameters

Due to the fact that the classification scores regarding the *Experiments I-III* were relatively low, we decided in this experiment to learn individual parameters from each corpus $\mathcal{C} \in \{\mathcal{C}_{DE}, \mathcal{C}_{DR}, \mathcal{C}_{EE}, \mathcal{C}_{EN}, \mathcal{C}_{GR}, \mathcal{C}_{SP}\}$ and to thereby improve the classification results.

Table 4. Top five performing feature set combinations.

F_{comb}	**Accuracy**
$\{F_1, F_2, F_5, F_{12}, F_{14}\}$	$61,8\%$
$\{F_3, F_{12}, F_{13}\}$	$61,62\%$
$\{F_1, F_2, F_5, F_{13}, F_{14}\}$	$61,62\%$
$\{F_2, F_5, F_{12}, F_{13}, F_{14}\}$	$61,26\%$
$\{F_1, F_3, F_{12}\}$	$61,26\%$

For this we first applied VEBAV on each $\mathcal{C}$ to obtain individual λ scores. Since there where six corpora, we constructed six tables, where the rows denote $F_1, F_2, ..., F_{14}$ and the coloumns the 14 λ values. Then, we picked those six tuples (F_i, λ_j), which led to the maximum accuracy score in each table. These tuples are listed in Table 5. One

Table 5. Most promising tuple for each individual training subcorpus.

$\mathcal{C}$	$(\mathcal{F}_i, \lambda_j)$	**Accuracy**
$\mathcal{C}_{DE}$	$(F_{12}, 0.6)$	$73,68\%$
$\mathcal{C}_{DR}$	$(F_{12}, 0.8)$	$60,76\%$
$\mathcal{C}_{EE}$	$(F_{12}, 1)$	$63,75\%$
$\mathcal{C}_{EN}$	$(F_3, 0.6)$	75%
$\mathcal{C}_{GR}$	$(F_3, 0.2)$	65%
$\mathcal{C}_{SP}$	$(F_7, 1)$	$71,25\%$

can see here that it definitely make sense to use corpus-dependent (or more precisely language/genre-dependent) parameters, instead of using a global setting. However, the price for better results may be expensive in terms of overfitting.

5.6 Evaluation results for the test set

In this section we evaluate VEBAV on the test set $\mathcal{C}_{PAN\text{-}Test}$, where we used all relevant information, learned from the prior experiments. For a better overview we divide the evaluations into three subsections, where we first show how VEBAV performed with generalized parameters then how it performed with individual parameters and finally how it performed regarding the runtime.

Evaluation results regarding generalized parameters. For the first evaluation we set as an input for VEBAV the generalized parameters $\lambda = 0.6$ and $\mathcal{F}_{12}$ that were learned from *Experiments I-II*. The results are given in Table 6. As can be observed from this table, using generalized parameters seems not to be the best choice, as achieving optimal results was possible for only one subcorpus, while for two subcorpora the results where even lower than a random guess. Moreover, it can be seen that the $AUC \cdot c@1$ scores are very low, which are not only caused by the low accuracies themselves, but

Table 6. Results regarding $\mathcal{C}_{\textit{PAN-Test}}$, using generalized parameters.

$\mathcal{C}$	**Accuracy**	$\boldsymbol{AUC \cdot c@1}$
$\mathcal{C}_{DE}$	80%	$0,40248$
$\mathcal{C}_{DR}$	45%	$0,2445525$
$\mathcal{C}_{EE}$	65%	$0,3147625$
$\mathcal{C}_{EN}$	45%	$0,2309625$
$\mathcal{C}_{GR}$	50%	$0,262025$
$\mathcal{C}_{SP}$	55%	$0,2602875$

also by an inappropriate calculation of ρ, performed through linear scaling. However, further investigations with other scaling methods must show if this hypothesis is valid.

Evaluation results regarding individual parameters. In the second evaluation we used the individual parameters, learned in *Experiment IV*. The results are given in Table 7. By looking on the results in this table, we can conclude once again that using an individual parameter setting over a global setting is much more promising. Thus, individual parameter settings should be (among other things) the subject for further investigations, when applying VEBAV on other corpora beyond those of PAN.

Table 7. Results regarding $\mathcal{C}_{\textit{PAN-Test}}$, using individual parameters.

$\mathcal{C}$	$(\boldsymbol{\mathcal{F}_i}, \lambda_j)$	**Accuracy**	$\boldsymbol{AUC \cdot c@1}$
$\mathcal{C}_{DE}$	$(F_{12}, 0.6)$	80%	$0,40248$
$\mathcal{C}_{DR}$	$(F_{12}, 0.8)$	55%	$0,30569$
$\mathcal{C}_{EE}$	$(F_{12}, 1)$	55%	$0,25801125$
$\mathcal{C}_{EN}$	$(F_3, 0.6)$	80%	$0,4146$
$\mathcal{C}_{GR}$	$(F_3, 0.2)$	70%	$0,362425$
$\mathcal{C}_{SP}$	$(F_7, 1)$	55%	$0,28072275$

Evaluation results regarding runtime. In the third and last evaluation we applied the entire PAN corpus $\mathcal{C}_{\textit{PAN-14}} = \mathcal{C}_{\textit{PAN-Train}} \cup \mathcal{C}_{\textit{PAN-Test}}$ on VEBAV, where we considered only the runtime needed for the clasification, rather than the clasification results. Regarding this evaluation we used a laptop with the following configuration: Intel® Core™ i5-3360M processor (2.80 GHz), 16GB RAM, 256GB SSD hard drive.

The runtime (measured in milliseconds) for each feature set and each subcorpus is given in Table 8. As can be seen in the *Median* column in Table 8, the fastest classification was performed with the feature set F_3 (*punctuation marks*). The reason for this is that F_3 leads to very few feature-lookups ($\approx$ 20 per document), such that IO-accesses are negligible. In contrast to this, F_1 leads to the highest runtime (excepting F_{12} and F_{13} since they are mixtures of existing sets), due to the fact that it requires an iteration over each character in a text. Another interesting observation is that token-based features (F_{8-11}) also require very low runtime. This is explained by the fact that there are much less tokens in texts than characters and thus, the number of lookups is also very limited.

Table 8. Runtime needed by `VEBAV` for the classification of $\mathcal{C}_{PAN\text{-}14}$ (considering each subcorpus).

	$\mathcal{C}_{DE}$	$\mathcal{C}_{DR}$	$\mathcal{C}_{EE}$	$\mathcal{C}_{EN}$	$\mathcal{C}_{GR}$	$\mathcal{C}_{SP}$	**Median**
F_1	$1,015$	148	$5,361$	$6,358$	$7,279$	$7,395$	$5,860$
F_2	775	113	$4,083$	$5,891$	$5,761$	$5,819$	$4,922$
F_3	44	7	235	359	150	307	193
F_4	55	6	358	623	445	502	402
F_5	98	6	396	675	470	569	433
F_6	356	17	$2,083$	$2,963$	$3,019$	$4,457$	$2,523$
F_7	532	29	$3,349$	$4,612$	$4,512$	$6,942$	$3,931$
F_8	67	8	402	428	371	530	387
F_9	70	8	449	478	438	582	444
F_{10}	53	10	253	283	253	328	253
F_{11}	95	18	458	521	481	583	470
F_{12}	$1,529$	178	$8,505$	$11,554$	$12,383$	$14,821$	$10,030$
F_{13}	$2,035$	287	$11,082$	$13,476$	$16,369$	$15,817$	$12,279$
F_{14}	269	26	$1,698$	$2,255$	$1,856$	$2,280$	$1,777$
Median	184	18	$1,078$	$1,465$	$1,169$	$1,432$	

The overall runtime for $\mathcal{C}_{PAN\text{-}14}$, without considering the subcorpora, is given in Figure 4. An interesting observation that can be made by comparing Figure 4 to Fig-

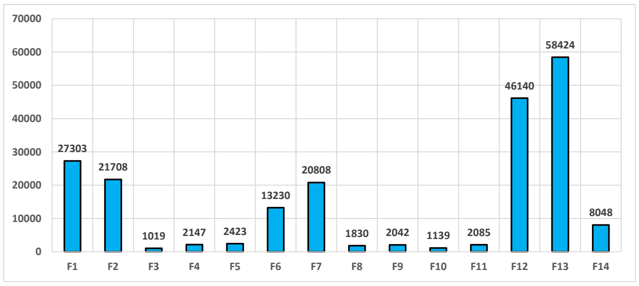


Figure 4. Runtime needed by `VEBAV` for the classification of $\mathcal{C}_{PAN\text{-}14}$ (without considering the subcorpora).

ure 2 is that F_5 can be seen as an optimal candidate when a trade-off between a high classification result and low runtime must be taken into account. Here, F_5 achieves a classification result of $59,96\%$ by requiring only 2,423 milliseconds. F_8 behaves in a similar way ($55,56\%$ accuracy by requiring only 1,830 milliseconds).

6 Conclusion & future work

In this paper we presented a simple, scalable and fast authorship verification scheme for the Author Identification task within the PAN-2014 competition. Our method provides a number of benefits as for example that it is able to handle (even very short) texts written in several languages, across different kinds of genre. Besides this, the method is independent of linguistic resources such as ontologies, thesauruses, language models, etc. A further benefit is the low runtime of the method, since there is no need for deep linguistic processing like POS-tagging, chunking or parsing. Another benefit is that the involved components within the method can be replaced easily as for example the distance function (required for classification), the acceptance-threshold or the feature sets including their parameters. Moreover, the classification itself can be modified easily, e.g. by using an ensemble of several distance functions.

Unfortunately, besides benefits our approach has several pitfalls too. One of the biggest challenges, for example, is the inscrutability of the methods parameter-space, due to the fact that the number of possible configuration settings is near infinite. Such settings include for instance the λ parameter for the involved distance function, the values for the n and k parameters, the weight (ω) and tolerance (τ) parameters that influence the classification quality but also other options such as ℓ (the number of chunks) or the used feature normalization strategy. Due to the complexity of our scheme we could only perform a small number of experiments to obtain at least an optimal λ and the most promising feature sets. This, however, was done only for the official PAN corpus but not for other corpora. Thus, it had not been proved if the learned parameters are able to perform satisfactorily on other corpora. Another challenge that remains unsolved is how to optimize the probability score ρ, determined in the decision determination step, as this value has also a strong influence on the resulting $AUC \cdot c@1$ scores, which in our case are very low. Hence, further tests are essential for the applicability of our scheme.

Acknowledgments. This work was supported by the CASED Center for Advanced Security Research Darmstadt, Germany funded by the German state government of Hesse under the LOEWE program (www.CASED.de).

References

1. Cha, S.H.: Comprehensive survey on distance/similarity measures between probability density functions. International Journal of Mathematical Models and Methods in Applied Sciences 1(4), 300–307 (2007)
2. Juola, P., Stamatatos, E.: Overview of the Author Identification Task at PAN 2013. In P. Forner, R. Navigli, and D. Tufis (eds) CLEF 2013 Evaluation Labs and Workshop - Working Notes Papers (2013)
3. Koppel, M., Schler, J.: Authorship Verification as a One-Class Classification Problem. In: Proceedings of the twenty-first international conference on Machine learning. pp. 62–. ICML '04, ACM, New York, NY, USA (2004)
4. Stamatatos, E.: A Survey of Modern Authorship Attribution Methods. J. Am. Soc. Inf. Sci. Technol. 60(3), 538–556 (Mar 2009)
5. Tax, D.M.J.: One-Class Classification. Concept Learning In the Absence of Counter-Examples. Ph.D. thesis, Delft University of Technology (2001)